UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Mejora de las predicciones en muestras desbalanceadas

Autora: Blanca Abella Miravet
Tutora: Ana María González Marcos

junio 2021

Blanca Abella Miravet
Mejora de las predicciones en muestras desbalanceadas

Blanca Abella Miravet
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

Agradecimientos

A mi familia y a mis amigos, por darme su apoyo incondicional y haber confiado en mí siempre.

"Get the right people. Then no matter what all else you might do wrong after that, the people will save you". Tom DeMarcos

Resumen

La clasificación es una de las técnicas de aprendizaje supervisado para realizar análisis predictivos con el resultado categórico, puede ser una clase binaria o multiclase. Hoy en día, hay mucha investigación y casos sobre clasificación utilizando varios algoritmos, desde un básico hasta avanzado como regresión logística, árbol de decisión, bosque aleatorio, etc. Han sido bien desarrollados y aplicado con éxito a muchos dominios. Sin embargo, la distribución de clases desequilibrada de un conjunto de datos ha encontrado serias dificultades para la mayoría de los algoritmos de aprendizaje de clasificadores que asumen una distribución relativamente equilibrada [1].

En este trabajo se presentan distintas técnicas para abordar este problema y conseguir unos resultados mejores. Estas técnicas son el remuestreo del conjunto de entrenamiento en los modelos y modificaciones en los algoritmos que usaremos para clasificar de manera que tengan en cuenta el desbalanceo presente. Aplicaremos estas técnicas sobre un conjunto de datos de clientes de una aseguradora a los que se ofrecerá contratar un seguro de coche con la misma.

Por otro lado, la clasificación de datos desbalanceados nos hace preguntarnos cuál es la mejor manera de evaluar un clasificador. Introducimos distintas métricas de evaluación que serán de gran utilidad para concluir si un clasificador está resolviendo el problema que tenemos.

Palabras clave

Clasificación, desbalanceo de clases, técnicas de remuestreo, métricas de evaluación

Abstract

Classification is one of the supervised *machine-learning* techniques to perform predictive analysis with the categorical result, it can be a binary or multiclass class. Today, there is a lot of research and cases on classification using various algorithms, from basic to advanced such as logistic regression, decision tree, random forest, etc. These have been well developed and successfully applied in many domains. However, the unbalanced class distribution of a data set has encountered serious difficulties for most classifier learning algorithms that assume a relatively balanced distribution cite Res.

In this paper, different techniques are presented to tackle this problem and achieve better results. These techniques are the resampling of the training set in the models and modifications in the algorithms that we will use to classify in a way that takes into account the present imbalance. We will apply these techniques on a set of customer data of an insurer who will be offered to contract a car insurance.

Moreover, the classification of unbalanced data makes us wonder which is the best way to evaluate a classifier. We introduce different evaluation metrics that will be very useful to conclude if a classifier is solving the problem we have.

Keywords

Clasification, class imbalance, resampling methods, evaluation metrics

Índice

Listas

Lista de figuras

Lista de tablas

1

Introducción

1.1. Motivación

La presencia de clases desbalanceadas en bases de datos supone un problema en los modelos predictivos ya que estos tienden a centrar su atención sobre los casos de la clase mayoritaria obteniéndose resultados que aparentemente son buenos, pero en definitiva es como predecir todo como la clase mayoritaria. El análisis de datos desbalanceados presenta una serie de características peculiares que merecen un trato distinto tanto a la hora de entrenar como a la hora de evaluar.

1.2. Objetivos

Este trabajo tiene como fin analizar los problemas que nos encontramos en la clasificación de clases de datos desbalanceadas, ver cuáles son las causas de esta mala clasificación y estudiar técnicas que ayuden a mejorarla.

Para aplicar la teoría utilizaremos una base de datos en la que exista un desbalanceo. En esta ocasión hemos optado por los datos de una compañía de seguros de vida que además ofrece un seguro de coche. El conjunto de datos lo hemos obtenido del repositorio *Kaggle* [2]. Analizaremos la base de datos y la estudiaremos para comprobar que se puede utilizar para la clasificación. Veremos que existe distintas maneras de enfrentarnos a este problema e iremos probando cada una para ver como afecta a la clasificación. Finalmente compararemos los resultados obtenidos para ver cómo se ha intentado solucionar el tema.

1.3. Estructura del documento

EL documento se divide en 3 partes: *marco teórico*, *experimentación* y *conclusiones*. En primer lugar se introducirá la teoría que abarca el problema que nos ocupa. Para ello hablaremos de la clasificación y los distintos algoritmos que en este trabajo se van a emplear: árbol de decisión, regresión

logística, *random forest* y *adaboost*. También se abordarán otros conceptos como las distintas métricas que se pueden emplear para evaluar un modelo de clasificación. Después se introducirá el concepto de desbalanceo de clases y se verán sus causas y distintas formas que existen para tratar de solucionarlo.

La parte de experimentación constará de un análisis de la base de datos que vamos a utilizar de ejemplo y los resultados obtenidos tras aplicar a los distintos clasificadores las técnicas correspondientes mencionadas en la teoría.

Para finalizar se hará una recapitulación de todo lo aprendido durante el trabajo en el apartado de conclusiones y se dará una visión de posible trabajo futuro.

2

Clasificación de datos

2.1. *Machine-Learning*

El *Machine-Learning* (aprendizaje automático) es una disciplina del campo de la Inteligencia Artificial que, a través de algoritmos, dota a los ordenadores de la capacidad de identificar patrones en datos masivos para hacer predicciones. Este aprendizaje permite a los computadores realizar tareas específicas de forma autónoma, es decir, sin necesidad de ser programados. [3]

El objetivo del *Machine-Learning* es crear un modelo que nos permita resolver una tarea dada. Luego se entrena el modelo usando gran cantidad de datos. El modelo aprende de estos datos y es capaz de hacer predicciones. La elección del mejor algoritmo de aprendizaje para un problema dado no es una tarea fácil, y dependerá del tipo de problema al que nos enfrentemos (clasificación, regresión, *clustering*,...).

Los algoritmos de *Machine-Learning* se dividen en tres categorías, siendo las dos primeras las más comunes:

Aprendizaje supervisado. Los algoritmos trabajan con datos "etiquetados" (*labeled data*), intentando encontrar una función que, dadas las variables de entrada (*input data*), les asigne la etiqueta de salida adecuada. El algoritmo se entrena con un conjunto de datos etiquetados y así "aprende" a asignar la etiqueta de salida adecuada a un nuevo valor, es decir, predice el valor de salida. El aprendizaje supervisado suele emplearse en problemas de clasificación y regresión. La diferencia entre ambos es que la variable objetivo es de tipo discreto en el caso de clasificación, y continuo en la regresión.

Aprendizaje no supervisado. Tiene lugar cuando no se dispone de datos etiquetados para el entrenamiento. Solo se conocen los datos de entrada, pero no existen datos de salida que correspondan con un *input* dado. Por tanto, no pretenden ajustar pares entrada-salida, sino que sirven para aumentar el conocimiento estructural de datos disponibles (o de posibles datos futuros). Estas tareas son por ejemplo, dar una agrupación de los datos según su similitud (*clustering*), simplificar la estructura de los mismos manteniendo sus características fundamentales (como en los procesos de reducción de la dimensionalidad), o extraer la estructura interna con la que se distribuyen los datos en su espacio original (aprendizaje topológico). En otras palabras, se trata de crear grupos homogéneos con la máxima heterogeneidad entre ellos.

La mayor parte de las definiciones, resultados teóricos, y algoritmos clásicos más importantes, se clasifican como algoritmos supervisados y, sobre todo en el pasado, muchos de los algoritmos no supervisados se reservaban para tareas de preprocesamiento de datos integrados en metodologías más amplias. Este hecho se debe, principalmente, a que el objetivo que dirige el aprendizaje supervisado está definido de una manera mucho más clara, y el no supervisado resulta más etéreo y difuso.

Sin embargo, no todos los algoritmos de *Machine-Learning* se pueden clasificar en una de las dos categorías anteriores. Estos otros irían en lo que se llama **aprendizaje por refuerzo**. Basándose en la psicología conductista, este tipo de aprendizaje se basa en mejorar la respuesta del modelo usando un proceso de retroalimentación. Su información de entrada es el *feedback* o retroalimentación que obtiene del mundo exterior como respuesta a sus acciones. Por lo tanto, el sistema aprende a base de ensayo-error. No es un tipo de aprendizaje supervisado porque no se basa estrictamente en un conjunto de datos etiquetados, sino en la monitorización de la respuesta a las acciones tomadas. Tampoco es un aprendizaje no supervisado ya que, cuando modelamos a nuestro "aprendiz", sabemos de antemano cuál es la recompensa esperada. [4]

2.2. Clasificación

En este trabajo nos centramos en métodos de aprendizaje supervisado ya que nuestro objetivo será predecir la pertenencia a una clase a partir de un conjunto de datos etiquetado.
Como hemos visto antes, la clasificación consiste en establecer una regla para predecir la etiqueta de clase a la que pertenece una observación a partir de un conjunto de datos de entrenamiento cuyas categorías son conocidas. Los algoritmos de clasificación permiten abstraer información, llevándola a una representación adecuada para la toma de decisiones.
Dentro de la clasificación, podemos distinguir [5]:

Clasificación binaria. Se utiliza para predecir a cuál de dos clases o categorías pertenece una instancia de un conjunto de datos. Suelen asignarse los valores 0 y 1 para diferenciar las categorías (etiquetas). Algunos de los escenarios donde se aplica este tipo de clasificación son la detección de enfermedades, analizar comentarios en alguna red social (positivo/negativo) o retención de clientes.

Clasificación multi-clase. Se utiliza para predecir a qué clase o categoría pertenece una instancia de un conjunto de datos. En este caso existen más de dos categorías para clasificar. Ejemplos de aplicación de este tipo de clasificación serían determinar la raza de un perro, reconocimiento de caracteres en pictogramas o categorización de comentarios en función del tema del que hablan.

Existen distintos tipos de algoritmos que son capaces de clasificar. Entre ellos veremos el funcionamiento de los árboles de decisión, la regresión logística y los llamados métodos conjuntos o multiclasificadores.

2.2.1. Modelos de clasificación

El objetivo de un modelo de clasificación es predecir la clase de futuras instancias o datos que no conoce. No podemos fiarnos únicamente de la información respecto al nivel de acierto del modelo ya que cuenta con toda la información desde el principio y no será representativo para saber cómo actuará con datos que no conoce. Por eso, cuando queremos aplicar un método de clasificación a un conjunto de datos, debemos dividir los datos en, al menos, dos subconjuntos: **entrenamiento** y **test**. Los datos de entrenamiento se emplearán para ir modificando los parámetros del algoritmo de acuerdo con las características concretas de las instancias de este conjunto teniendo en cuenta su etiqueta de clase. Una vez se haya entrenado el modelo, se prueba con los datos del conjunto de test para medir su rendimiento. No obstante, el resultado de este procedimiento puede variar en función de la partición que hagamos sobre el conjunto de datos original. Por esta razón se han diseñado técnicas de validación para comprobar si las particiones realizadas para separar los conjuntos de entrenamiento y test están afectando a la creación del modelo.

Por otra parte, también necesitamos cerciorarnos de que el modelo es capaz de generalizar bien, es decir, que sea capaz de reconocer datos que difieran un poco de los que ha entrenado. En ocasiones un modelo se ajusta tanto a los detalles de las características de los datos de entrenamiento que no es capaz de clasificar correctamente otros datos. Este fenómeno se denomina *overfitting* (sobreajuste). Para evitarlo se suelen ajustar algunos parámetros de los clasificadores como, por ejemplo, la profundidad en el caso de árboles de decisión o *Random Forest.*

Validación cruzada

Uno de los métodos de validación más empleados es la validación cruzada de k particiones. Esta técnica consiste en dividir los datos de manera aleatoria en k grupos del mismo tamaño. $k-1$ de esos grupos serán utilizados como conjunto de entrenamiento y el grupo restante será empleado para realizar la evaluación. Este proceso se repite k veces utilizando un grupo distinto como validación en cada iteración. El método genera k estimaciones cuyo promedio se emplea como estimación final [6]. Como hemos explicado, esto sirve para confirmar que el resultado obtenido es independiente de los datos concretos con los que se ha realizado la prueba de clasificación y que, por tanto, el valor no es fortuito. La figura 2.1 muestra el proceso seguido por este método.

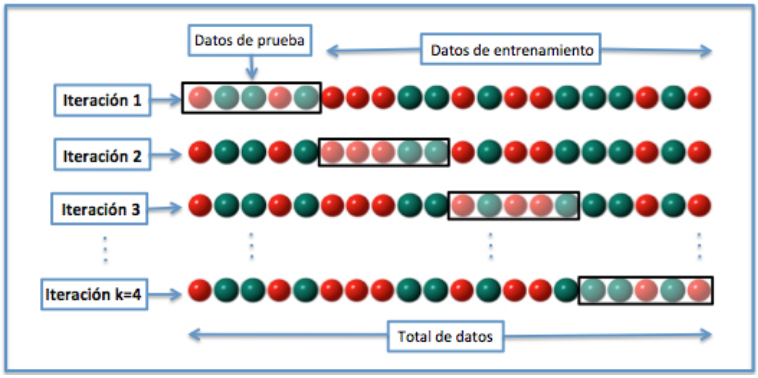


Figura 2.1: Validación cruzada de k particiones
Fuente: `https://es.wikipedia.org/wiki/Validación_cruzada`

2.3. Clasificadores

En esta sección analizaremos más en profundidad algunos de los algoritmos de clasificación que se han mencionado anteriormente. Es importante indicar que no todos los métodos de clasificación son igual de útiles para todos los conjuntos de datos. En la práctica, se recomienda probar y comparar el comportamiento de diferentes algoritmos para escoger el más adecuado en cada caso.

2.3.1. Árboles de decisión

Un árbol de decisión es un método de clasificación predictivo en el que se agrupan bajo la misma etiqueta observaciones que tienen características parecidas para la misma etiqueta. Para ello, se establecen una serie de reglas o decisiones que irán determinando los subgrupos en los que se dividirá el conjunto. En los árboles de decisión encontramos los siguientes componentes: nodos, ramas y hojas. Los nodos representan las variables de entrada, es decir, las observaciones con las que contamos para entrenar; las ramas determinan una respuesta ante una decisión o característica diferenciadora y las hojas son los posibles valores de la variable de salida, es decir, la clasificación final. Llamamos nodo raíz al nodo desde el que se hace la primera división. El número máximo de nodos bajo el nodo raíz determina el nivel de profundidad del árbol [7]. Como se ha mencionado, es posible ajustar esta profundidad para evitar un sobreajuste en el entrenamiento de los datos. En la figura 2.2 podemos observar el funcionamiento de un árbol de decisión.

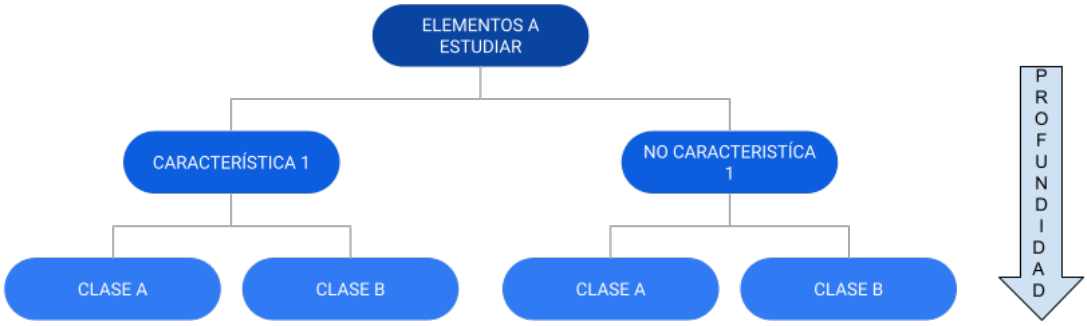


Figura 2.2: Representación de un árbol de decisión

2.3.2. Regresión logística

La regresión logística es un método de clasificación que aplica técnicas estadísticas para medir la relación de la etiqueta de clase con el resto de características de la muestra [8]. La regresión logística modela la probabilidad de pertenecer a cada clase mediante la ecuación:

$$p_i = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_{1,i} + ... + \beta_k x_{k,i})}},$$

donde cada x representa una de las características a analizar y β el "peso"de cada una. Podemos observar que lo que realmente se predice es una probabilidad para cada instancia. Para convertirlo en una clasificación se fija un umbral de probabilidad a partir del cual pertenece a una u otra clase. La figura 5.5 nos muestra como se modela el algoritmo en función de los datos analizados.

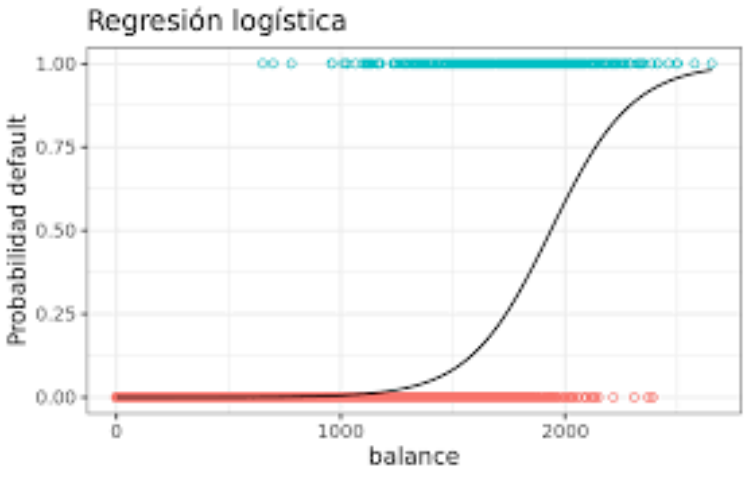


Figura 2.3: Regresión logística
Fuente: `https://www.cienciadedatos.net/documentos/27_regresion_logistica_simple_y_multiple`

2.3.3. Métodos conjuntos

Los sistemas basados en métodos conjuntos se obtienen combinando varios clasificadores simples. Por esta razón, también se les conoce como **multiclasificadores**. Para entender la importancia y la utilidad de este tipo de clasificador, podemos pensar que es algo que nosotros mismos usamos en la vida real. Cuando vamos a tomar una decisión, preguntamos y consultamos distintas fuentes. Por ejemplo, al acceder a cierto tratamiento médico, comprar un coche... Al final, la decisión se hace combinando toda la información recogida aplicando varios criterios (varios clasificadores) y valorando cuál se ajusta mejor a nuestro interés. Los multiclasificadores realizan este mismo proceso.
El objetivo de este tipo de métodos es combinar las predicciones de varios clasificadores para así construir uno nuevo que sea más robusto y preciso. La idea básica detrás de esto es construir distintos "expertos" y dejarles decidir. Este método permite conseguir mejores resultados en la predicción de clases que un único clasificador.

	María	**compró**	**un**	**libro**	**Prob. Acierto**
C1	sust	verbo	sust	sust	75 %
C2	verbo	verbo	det	sust	75 %
C3	sust	adjetivo	det	adverbio	50 %
Mayoría	sust	verbo	det	sust	100 %

Tabla 2.1: Aplicación multiclasificadores

En la Tabla 2.1 presentamos un ejemplo de multiclasificador. El problema es clasificar las palabras de una frase según su función. Se aplican tres clasificadores, C1, C2, C3. Los clasificadores C1,C2 y C3 tienen una tasa de acierto del 75 %,75 % y 50 % respectivamente. Sin embargo, un multiclasificador como representa la última fila de la Tabla 2.1 que clasifica en función de lo que la mayoría de estos clasificadores individuales determina, consigue un acierto del 100 %.

Aunque como idea parezca simple, en la práctica aparecen ciertas complicaciones a la hora de crear modelos multiclasificadores. Conseguir encontrar clasificadores independientes entre sí y que sean precisos puede ser muy complejo.

A continuación vamos a explicar algunos de los multiclasificadores más empleados [9].

Bagging

El *bagging* es un método de clasificación donde varios clasificadores simples son aplicados en paralelo. El principal objetivo es utilizar la independencia entre ellos para reducir el error global. La forma de conseguir que los errores se vean compensados, y por tanto reducidos, es entrenar cada modelo con subconjuntos aleatorios (con remplazo) del conjunto de entrenamiento. Una vez entrenados los modelos, se pasa a la clasificación con el conjunto de prueba. Cada una de las nuevas instancias será

clasificada por cada uno de los modelos y cada una de las predicciones será tenida en cuenta como si fuese un voto. Finalmente, la clase con mayor número de votos es la que el método *bagging* predecirá. El número de clasificadores a entrenar dependerá de las características concretas de cada conjunto de datos.

El *Random Forest* (bosque aleatorio) es un caso particular de *bagging*. Es uno de los métodos de clasificación más utilizados debido a su simplicidad y sus buenos resultados. Se basa fundamentalmente en el modelo de árboles de decisión. Consiste, como su nombre deja intuir, en una colección de árboles de decisión individuales que trabajan como un conjunto. Cada uno de ellos entrena un subconjunto de los datos disponibles y se ocupa sólo de un subconjunto de las características. Esto provoca un gran abanico de posibilidades, ya que la variable que se selecciona como óptima para la división, varía en cada árbol al depender del resto del subconjunto. La decisión final se toma como una media de las decisiones individuales de los árboles.

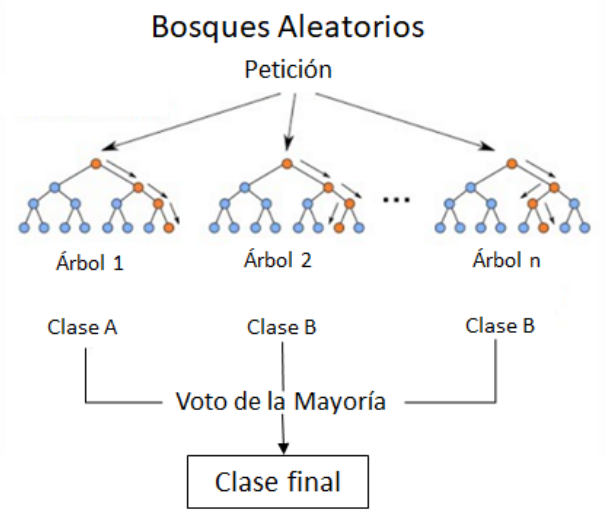


Figura 2.4: Funcionamiento Random Forest
Fuente: https://medium.com/datos-y-ciencia

En la figura 2.4 podemos apreciar el proceso que sigue el *Random Forest*. Vemos que consta de n árboles de decisión independientes que determinan una clase y finalmente se opta por la clase que se ha predicho un mayor número de veces. Como hemos visto en el caso del árbol de decisión, se puede ajustar la profundidad de los árboles para evitar el sobreajuste. También el número de árboles sirve para combatir este problema.

Boosting

El *boosting*, a diferencia del bagging, combina los clasificadores de manera secuencial. La idea es mejorar la predicción entrenando secuencialmente los clasificadores de manera que cada uno trate de mejorar los errores cometidos por el clasificador anterior. Uno de los algoritmos de *boosting* más

utilizados es el llamado *Adaboost* (*Adaptive boosting*). El algoritmo base con el que se suele emplear Adaboost es un árbol de decisión binarios con solo un nivel de profundidad (*stumps*). La idea principal de este algoritmo es que cada *stump* se vea afectado por el *stump* anterior. La manera en la que un *stump* interviene en el peso del siguiente depende del error cometido en su clasificación y así sucesivamente [10].

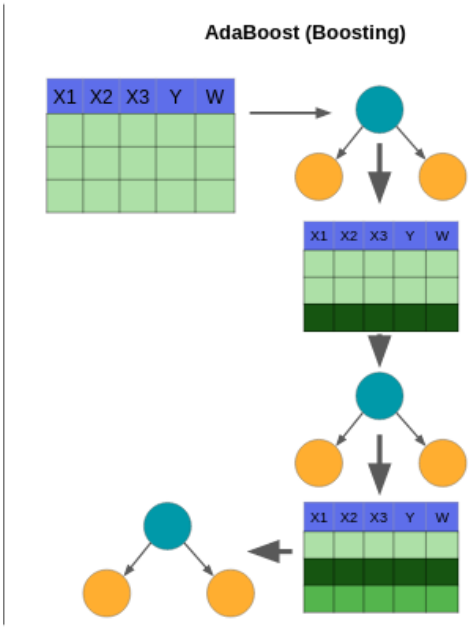


Figura 2.5: Funcionamiento AdaBoost
Fuente: `https://algotech.netlify.app/blog/xgboost/`

Bagging vs Boosting

Como hemos visto, la principal diferencia entre estos dos métodos de clasificación es que el bagging se realiza aplicando simultáneamente los clasificadores simples con distintas muestras mientras que el boosting aplica los modelos secuencialmente modificando el peso de los datos de la muestra que han sido mal clasificados la vez anterior. Además, podemos distinguir los problemas en las que es preferible usar uno u otro. Si el clasificador base que vamos a usar tiene problemas de overfitting, entonces el bagging nos será más útil para solucionarlo. Si el problema es que el clasificador base no es demasiado preciso, el boosting creará un modelo con menos errores.

3

Métricas de evaluación

En este capítulo veremos cómo evaluar la calidad de un modelo entrenado utilizando distintas métricas para cuantificar el rendimiento de este. Esto es importante para comparar distintos modelos y elegir el mejor. La manera más intuitiva de probar si un clasificador es bueno, es mirando el porcentaje de acierto en la clasificación de los datos de test. Sin embargo, veremos que el hecho de que este porcentaje sea alto o mayor que otros, no indica necesariamente que el clasificador sea óptimo.

La llamada **matriz de confusión**, es una herramienta que nos muestra el desempeño de un clasificador describiendo como se distribuyen los valores reales y nuestras predicciones [11].

		Predicciones	
		0	1
Observaciones	0	Verdaderos Negativos (VN)	Falsos Positivos (FP)
	1	Falsos Negativos (FN)	Verdaderos Positivos (VP)

Figura 3.1: Matriz de confusión

- Verdaderos negativos: Son los elementos pertenecientes a la clase negativa que han sido clasificados correctamente.
- Falsos positivos: Son los elementos que pertenecen a la clase negativa pero han sido clasificados como positivos.
- Falsos negativos: Son los elementos pertenecientes a la clase positiva que han sido clasificados como negativos.
- Verdaderos positivos: Son los elementos de la clase positiva que han sido bien clasificados.

La diagonal principal, contiene las predicciones correctas y la otra refleja los errores cometidos por el clasificador. Como hemos mencionado antes, no se deben tener en cuenta únicamente estos datos para valorar el buen funcionamiento de un clasificador. Como veremos más adelante, en problemas de clasificación con clases desbalanceadas un porcentaje alto de aciertos no implica que un clasificador sea bueno. La matriz de confusión no es una medida de rendimiento como tal, pero todas las métricas que veremos a continuación se basan en los datos que contiene.

Exactitud o Accuracy (AC)

Es la relación entre las predicciones correctas y el número total de predicciones, la tasa de acierto del clasificador.

$$AC = \frac{VP + VN}{VP + VN + FN + FP}$$

Esta métrica tiene la ventaja de que es muy fácilmente interpretable, sin embargo solo es recomendado medir un modelo con este valor si las clases están relativamente balanceadas ya que presenta limitaciones en problemas con clases desbalanceadas al haber muchos más datos de alguna de ellas.

Precisión (P)

La precisión mide el nivel de calidad del modelo. Representa el porcentaje de positivos que verdaderamente lo son.

$$P = \frac{VP}{VP + FP}$$

Esta métrica es muy interesante cuando el coste de los falsos positivos es muy elevado. Esto ocurre por ejemplo en la detección de correo *spam*. Que se 'cuele' un correo spam en la bandeja de entrada no es tan grave como que uno normal acabe en la carpeta de spam.

Sensibilidad o Recall (R)

Es la proporción de casos positivos que son correctamente clasificados, es decir, la capacidad del modelo de detectar una condición.

$$R = \frac{VP}{VP + FN}$$

La sensibilidad nos indica qué porcentaje de la clase está correctamente identificado. Nos interesará que este valor sea lo más alto posible en problemas como la detección de enfermedades, donde es importante detectar el máximo número de personas que la padezcan y no dejar ninguna sin atender.

En la práctica, hay que tratar que precisión y recall sean lo mayor posible, aunque es una tarea difícil ya que están relacionadas de manera que si se quiere aumentar la precisión disminuirá el recall y viceversa.

F_β-SCORE

Es una métrica muy empleada ya que nos resume la precisión y la sensibilidad en una sola. Es de gran utilidad en clases desbalanceadas y en casos en los que el "coste" de los falsos positivos y los falsos negativos es diferente.

$$F_\beta = (1+\beta^2)\frac{precision.recall}{(\beta^2.precision)+recall}$$

El valor de β dependerá de a qué queramos dar más importancia. Si consideramos que la sensibilidad es más importante, utilizaremos $\beta > 1$. Por el contrario, optaremos por β <1 si es la precisión lo que nos importa más. $\beta = 1$ indica que ambas nos interesan por igual.

Coeficiente de correlación de Matthews (MCC)

El coeficiente de Matthews es una métrica no tan conocida pero que tiene una gran importancia. Como el F1 score, es un valor que trata de dar una visión global acerca de la matriz de confusión. Este valor viene dado por la siguiente fórmula [12]:

$$MCC = \frac{VNxVP - FPxFN}{\sqrt{(VN+FN)(FP+VP)(VN+FP)(FN+VP)}}$$

Si nos fijamos en la ecuación y la comparamos con la del F-1 score, podemos observar que mientras el F-1 score no tiene en cuenta el recuento de verdaderos negativos, el MCC engloba las cuatro entradas de la matriz. Davide Chicco, autor de Diez consejos rápidos para el aprendizaje automático en biología computacional , comentó que el MCC "es alto solo si su clasificador está funcionando bien tanto en los elementos negativos como en los positivos".

Al ser un coeficiente de correlación, sus valores están comprendidos entre -1 y 1, siendo 0 el caso de un clasificador aleatorio.

Curva ROC

Una curva ROC es un gráfico que muestra el rendimiento de un modelo de clasificación. Relaciona la sensibilidad con el porcentaje de falsos positivos. Tiene sentido ya que si aumentamos la sensibilidad, el modelo será más optimista, es decir, clasificará más datos como positivos y por tanto, aumentarán los falsos positivos. La tasa de falsos positivos se calcula como

$$TFP = \frac{FP}{FP+VN}$$

En la figura 3.2, imaginemos una línea que va por la diagonal del [0,0] al [1,1]. Esta representaría un mal clasificador, es decir, aquel que no hace una mejor clasificación que la que podría hacerse de manera aleatoria o constante. La curva ROC de un buen modelo debe mantenerse lo más alejada posible de la línea diagonal dirigiéndose hacia la esquina superior izquierda (punto [0,1]). Para resumir la curva se utiliza el área bajo la curva (**AUC**). Su rango de valores va desde 0.5, valor que representa un modelo sin capacidad discriminante, hasta 1 que supone una clasificación perfecta. Por tanto, cuanto mayor sea el AUC, mejor será el modelo.

Curva P-R (precision-recall)

La curva PR es la gráfica resultante de relacionar la precisión y el recall de un modelo. Nos permite ver a partir de qué recall tenemos una degradación de la precisión. Un modelo perfecto sería aquel que pasase por el punto (1,1), por tanto, cuanto más se acerque a esa esquina mejor. Al igual que con la curva ROC, también se puede calcular el AUC de esta curva para comparar distintos modelos.

Es preferible usar esta curva en vez de la curva ROC en casos en los que exista un desbalanceo en las clases. Esto se debe a que la curva ROC puede dar una visión optimista del modelo por la dependencia de los falsos negativos que en conjuntos con clases desbalanceadas aumentará.

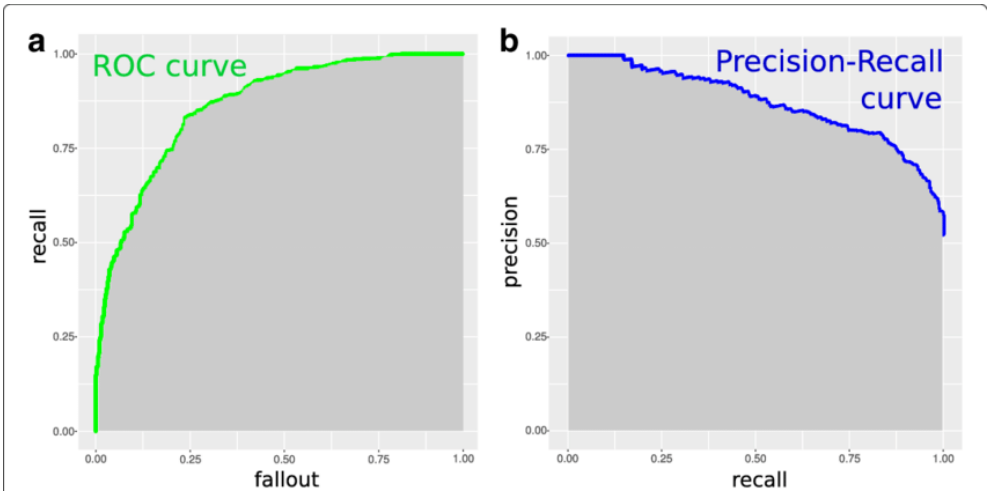


Figura 3.2: Curvas ROC y PR
Fuente: https://www.researchgate.net/figure/

4

El desbalanceo de clases

En los últimos años el análisis y la clasificación de grandes cantidades de datos ha cobrado una gran importancia. A medida que avanza la tecnología para clasificar, la cantidad de datos clasificables también lo hace. Así, la clasificación de datos se vuelve más difícil debido al gran tamaño de los datos y al desbalanceo natural de los datos. El desbalanceo viene dado por una desproporción en el número de elementos en cada clase dentro de un conjunto de datos. En estos casos, nos encontramos con una clase "minoritaria", o de la cual tenemos menos muestras, y otra/s "mayoritarias", o que están más representadas. El desbalanceo puede ser mayor o menor en función de cuántos elementos de la clase mayoritaria haya por cada uno de la minoritaria.

4.1. *¿Por qué supone un problema?*

Numerosos estudios coinciden en que la distribución desigual de las clases supone un problema a la hora de clasificar utilizando los algoritmos de clasificación que hemos visto hasta ahora ya que no manejan bien bases de datos con esta condición. Existen numerosas causas por las que esto sucede. Estos algoritmos asumen implícitamente que las clases están más o menos balanceadas; están basados en la precisión, es decir, su objetivo es minimizar el error total, con lo que la clase minoritaria influye poco y los datos de estas pueden ser tomados como ruido o datos atípicos; y asumen también que todos los errores tienen el mismo coste. Esto es un problema en conjuntos con desajuste entre los datos ya que, por lo general, son los datos pertenecientes a la clase minoritaria los que salen perjudicados de este tipo de técnicas, cuando justamente son estos los que más necesitan estar bien clasificados [13].

Tomemos por ejemplo un conjunto de datos con 990 muestras de cierta clase "A" y tan solo de 10 de otra clase "B". Un algoritmo aprenderá que la mejor suposición que puede hacer será que todo elemento es de la clase "A", dado que de este modo obtendrá una tasa de acierto del 99 %. Sin embargo, no podemos basarnos únicamente en este dato ya que esta suposición no es para nada acertada teniendo en cuenta que ha fallado en la predicción del 100 % de los elementos que eran

realidad de la clase “B”.

Este problema se suele ver mucho en áreas como la detección de enfermedades en donde tenemos bases de datos con miles de casos “negativos” de ciertas enfermedades frente a unos pocos “positivos”; de fraudes bancarios con pocas operaciones fraudulentas entre miles de operaciones no fraudulentas; de correos spam o en un funnel de marketing, por ejemplo.

4.2. *Métodos para abordar el problema del desbalanceo*

El problema del desbalanceo de clases se está abordando en la actualidad de forma activa y son muchos los investigadores que estudian y proponen nuevas técnicas para poder hacerle frente. La mayoría de las soluciones solo se han concentrado en resolver problemas de clasificación binaria. Un área de gran interés es, por ejemplo, la identificación de enfermedades raras dado un conjunto de síntomas. Para tratar el problema del desbalanceo, la comunidad investigadora suele optar principalmente por estas técnicas:

Preprocesado. Consiste en hacer un procesamiento de la muestra de datos antes del entrenamiento para minimizar el desajuste existente entre las clases.

Refinamieno de algoritmos. Ajustar los parametros de los clasificadores para que tengan en cuenta el desbalanceo existente y/o modificar la penalización de una mala clasificación.

Además, se pueden emplear técnicas que combinen los dos tipos y así mejorar el rendimiento de ambos.

En la práctica, se recomienda probar y comparar el comportamiento de distintas técnicas y escoger la más apropiada para cada caso, ya que no todos los conjuntos de datos se comportan igual ni tienen las mismas características.

4.2.1. *Técnicas de preprocesado*

Para enfrentarse al problema del desbalanceo de clases se aplican diversos métodos de remuestreo de los datos para cambiar la distribución de las clases conviertiendo la muestra en una muestra equilibrada. Podemos dividir estas técnicas en métodos de *oversampling* y de *undersampling* [14].

Oversampling

Este método pretende solucionar el problema del desbalanceo replicando datos de la clase minoritaria reduciendo así el desbalanceo. Existen varias maneras de aplicar este método, entre los que destacamos:

Random oversampling. Consiste en duplicar instancias de la clase minoritaria de manera aleatoria consiguiendo una paridad entre la cantidad de datos de las distintas clases.

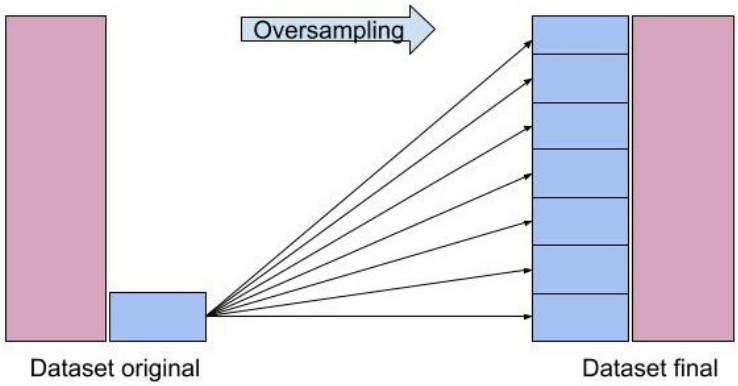


Figura 4.1: Random Oversampling

En la Figura 4.1 vemos que aplicando este método conseguimos equilibrar las clases, sin embargo, no conseguimos información nueva al modelo y podemos estar multiplicando instancias erróneas o ruido.

SMOTE. Mediante esta técnica se generan instancias nuevas de la clase minoritaria a partir de otras ya existentes. SMOTE selecciona un elemento de la clase minoritaria y de entre sus vecinos más cercanos selecciona uno y crea un nuevo dato entre ambos [15]. En la Figura 4.2 podemos ver este proceso.

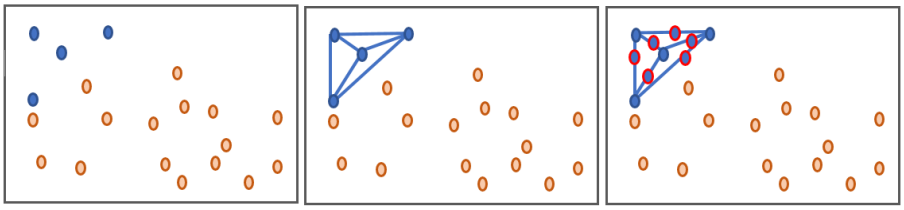


Figura 4.2: Funcionamiento SMOTE
Fuente: `https://datasciencecampus.github.io/balancing-data-with-smote/`

Undersampling

Con este método se reduce el número de observaciones de la clase mayoritaria para conseguir un conjunto de datos balanceado. Es recomendable usar este método cuando tenemos una muestra muy grande ya que reduciendo instancias mejoraremos el tiempo de entrenamiento del modelo. Como en el oversampling, destacamos dos maneras distintas de aplicarlo:

Random Undersampling. Se suprimen aleatoriamente instancias de la clase mayoritaria. Esta es una de las técnicas más antiguas utilizadas para aliviar el desequilibrio en el conjunto de datos. Sin embargo, puede aumentar la varianza del clasificador y potencialmente descartar muestras útiles. La Figura 4.3 nos muestra este método de manera gráfica.

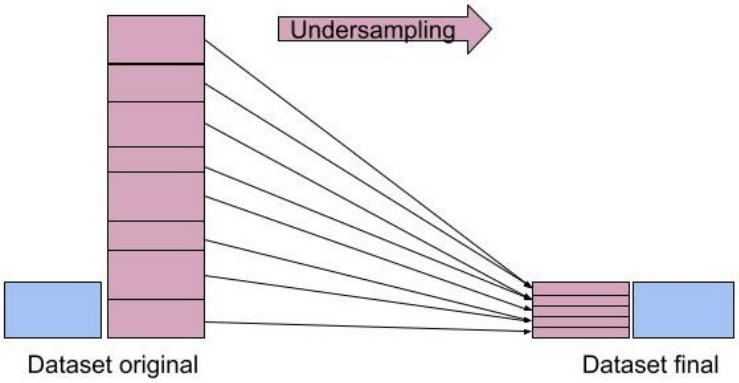


Figura 4.3: Random undersampling

Tomek Links. Un tomek link es un par de instancias de clases diferentes pero que son muy parecidas. El algoritmo Tomek busca este tipo de pares y elimina el dato perteneciente a la clase mayoritaria. Así se pueden establecer grupos bien definidos en el conjunto de entrenamiento y conducir a un mejor rendimiento de la clasificación [16]. Lo vemos en la Figura 4.4.

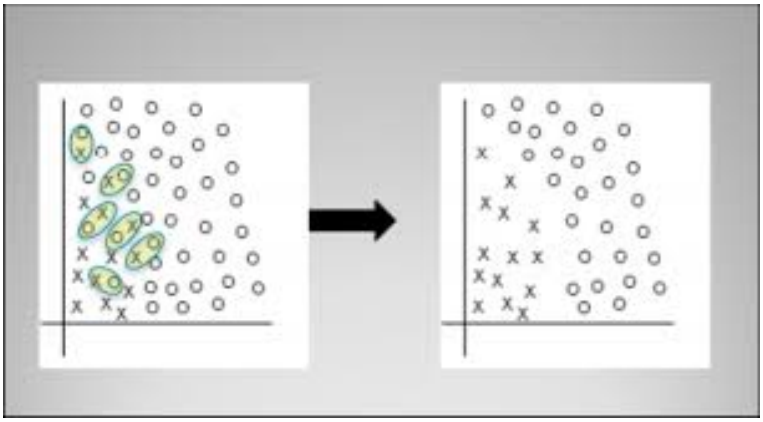


Figura 4.4: Funcionamiento Tomek Links
Fuente: `https://www.kdnuggets.com/2016/08/learning-from-imbalanced-classes.html/2`

4.2.2. Refinamiento de algoritmos

Otra manera de abordar el problema del desbalanceo es mediante el aprendizaje sensible al coste. Este método tiene en cuenta los costes de los distintos errores que se pueden cometer en una clasificación durante la construcción del modelo. El modelo se centra en minimizar el coste en vez de la tasa de error del clasificador, por lo que presta más atención a la clase minoritaria que suele ser la prioritaria. Existen varias maneras de ajustar los algoritmos para esto. La primera consiste en asignar pesos a las clases modificando así el conjunto de entrenamiento en función de los costes. Otra forma sería en modificar el funcionamiento interno de cada algoritmo para que tenga en cuenta los costes [17].

5 Experimentación

En este capítulo vamos a poder visualizar como las estrategias explicadas en las anteriores secciones tienen verdaderamente un efecto notable en la clasificación de bases de datos desbalanceadas. Primero vamos a ver de manera breve que instrumentos hemos utilizado para la aplicación de estos métodos.

5.1. Entorno

Para llevar a cabo la clasificación hemos optado por utilizar el lenguaje **python**. Python es uno de los lenguajes de programación que domina dentro del ámbito del *Machine-Learning*. Al tratarse de un software libre, muchos usuarios han podido implementar sus algoritmos dando lugar a numerosas librerías donde encontrar casi todas las técnicas de *Machine-Learning* existentes [18].

5.1.1. Librerías

Para la realización de esta clasificación nos hemos servido principalmente de dos librerías que poseen funciones muy útiles para llevarla a cabo.

Scikit-learn

Scikit-learn es una librería de código abierto que unifica bajo un único marco los principales algoritmos y funciones, facilitando en gran medida todas las etapas de preprocesado, entrenamiento, optimización y validación de modelos predictivos [18].

Imbalanced-learn

Imbalanced-learn es una librería de python que ofrece numerosos métodos de remuestreo comunmente utilizados en conjuntos de datos que presentan un desbalanceo entre sus clases. Además, esta librería es compatible con scikit-learn [19].

5.2. Análisis del conjunto de datos

Para ilustrar la aplicación de las técnicas descritas en el marco teórico a la clasificación, hemos optado por los datos de una compañía de seguros de vida que además ofrece un seguro de coche. El conjunto de datos lo hemos obtenido del repositorio de Kaggle [2]. La cuestión está en conseguir identificar qué personas aseguradas aceptarán el seguro de coche si se les ofrece. Según este problema, deberemos tener más en cuenta el recall del modelo que la precisión a la hora de tomar la decisión de cual es más útil. Esto es debido a que es preferible saber detectar lo mejor posible futuros clientes y no perderlos que ofrecer la posibilidad a clientes que realmente no aceptarán el seguro. Aún así haremos un análisis de varias métricas para ver como varían al aplicarse las técnicas.

Antes de entrenar un modelo predictivo, o incluso antes de realizar cualquier cálculo con un nuevo conjunto de datos, es muy importante realizar una exploración descriptiva de los mismos. Este proceso permite entender mejor qué información contiene cada variable, así como detectar posibles errores [18]. Primero vamos a ver qué estructura tiene la base de datos de la que disponemos para hacernos una idea de los atributos con los que contamos para la clasificación y comprobar si se puede trabajar con ella.

```
Estrucutura de los datos:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 382154 entries, 0 to 382153
Data columns (total 12 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   id                    382154 non-null  int64
 1   Gender                382154 non-null  object
 2   Age                   382154 non-null  int64
 3   Driving_License       382154 non-null  int64
 4   Region_Code           382154 non-null  float64
 5   Previously_Insured    382154 non-null  int64
 6   Vehicle_Age           382154 non-null  object
 7   Vehicle_Damage        382154 non-null  object
 8   Annual_Premium        382154 non-null  float64
 9   Policy_Sales_Channel  382154 non-null  float64
 10  Vintage               382154 non-null  int64
 11  Response              382154 non-null  int64
dtypes: float64(3), int64(6), object(3)
```

Figura 5.1: Información de la base de datos.

Como vemos en la Figura 5.1, contamos con un total de 382154 entradas con 12 variables cada una. Observamos que no existen valores nulos y que tenemos datos de tipo numérico y de tipo categórico. Vamos a modificar las variables categorías asignándoles un número para poder trabajar mejor con ellas. La variable *'id'* es simplemente un identificador. Comprobando que no existen duplicados, podemos deshacernos de ella para agilizar futuros procedimientos.

A continuación veremos en detalle algunas de las variables y haremos modificaciones en ella para mejorar el proceso de clasificación.

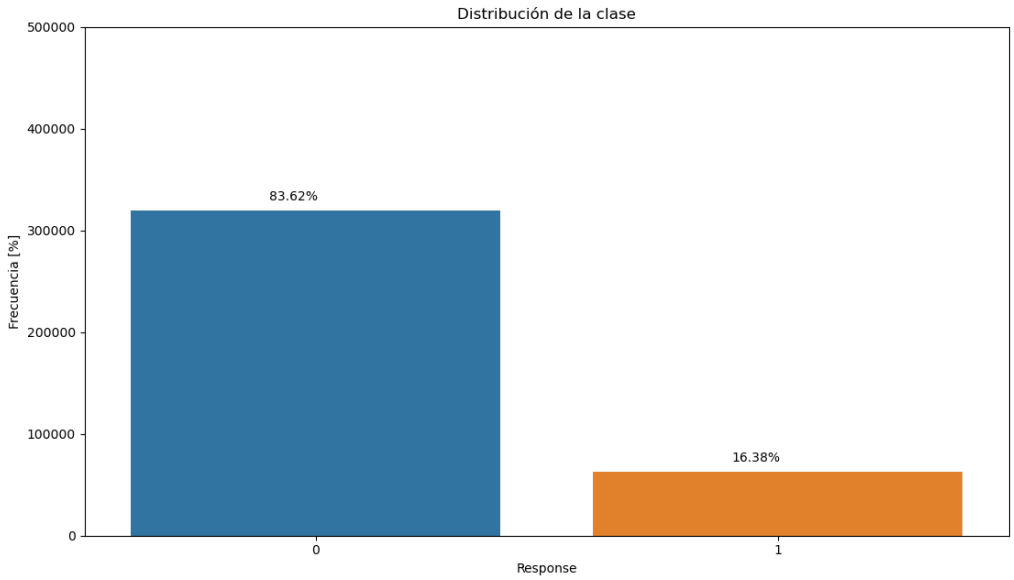


Figura 5.2: Gráfico del balanceo de clases de la base de datos

En la Figura 5.2 podemos ver como hay un claro desbalanceo entre las personas que deciden no contratar el seguro (azul) y las que finalmente optan por hacerlo (naranja). La base de datos es, por tanto, válida para el experimento que vamos a realizar.

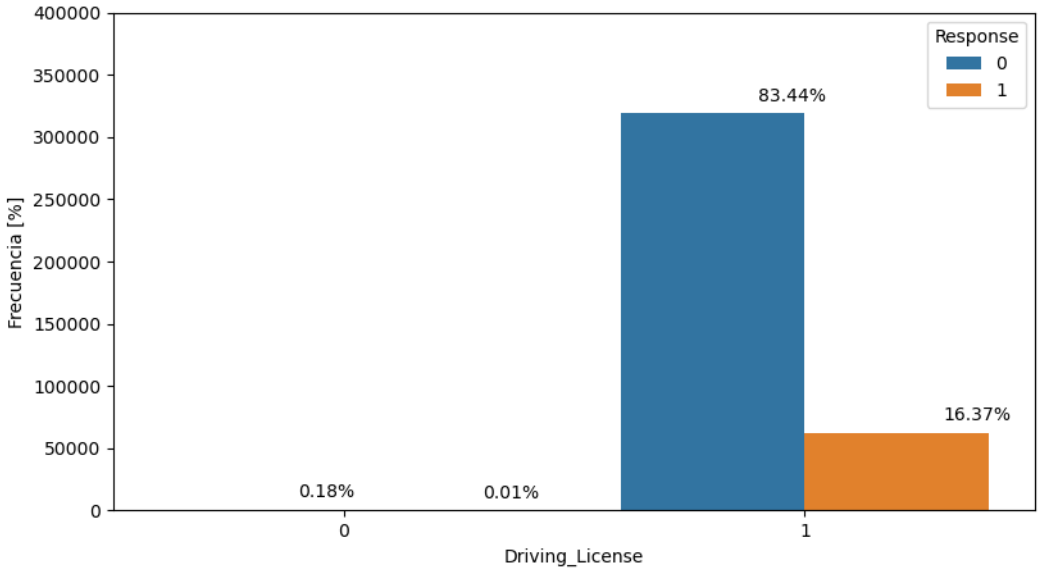


Figura 5.3: Respuesta según carnet de conducir

En la Figura 5.3 podemos apreciar que las personas que no tienen carnet de conducir no son ni un 0.2 % de la muestra y realmente solo un 0.01 % de las personas aseguradas no tienen carnet y deciden contratar el seguro. Este hecho puede ser perjudicial a la hora de entrenar nuestro modelo.

Al crear los conjuntos de validación cruzada puede ocurrir que, debido a su poca presencia, no haya ninguna observación con esa clase. Por tanto, decidimos prescindir de esta variable.

En la descripción de la base de datos se nos da a conocer que las variables correspondientes al canal por el que se ha ofrecido el seguro están codificados asignando un número a cada uno (diferentes agentes, teléfono, mail...). Vamos a ver su distribución, ya que aunque sea un valor numérico realmente no es una variable cuantitativa.

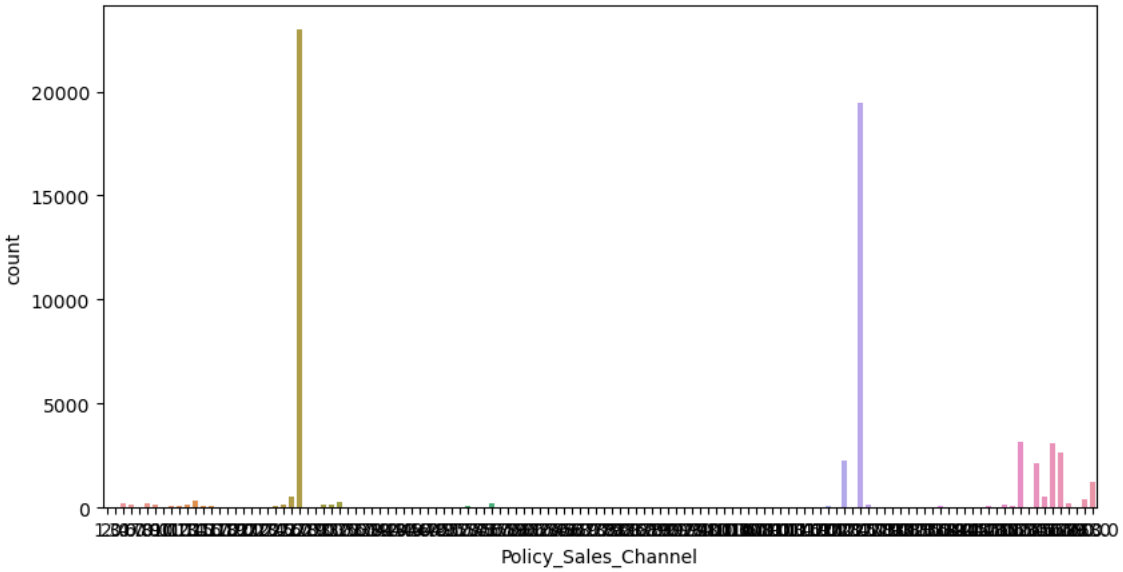


Figura 5.4: Canales de contratación

Podemos observar en la figura 5.4 que destacan dos canales de contratación por encima del resto. Por motivos similares al caso de deshacernos de la variable '$Driving\_License$' vamos a agrupar todos los datos que no pertenecen a los dos canales principales en uno solo. De esta manera, tendremos solo tres categorías para esta variable: si ha sido por cada uno de los canales más utilizados y el resto. Además, la convertiremos en variables dummies. Así cada una de las tres categorías pasará a ser una variable del modelo .

De la misma manera, vemos en la Figura 5.5 que hay una región donde la gente es mucho más propensa a contratar el seguro que en el resto. Seguimos el mismo procedimiento que con los canales. Convertiremos los posibles valores de la variable en dos, el principal y el resto.

Si observamos la matriz de correlación de la Figura 5.6 podemos concluir que ninguna de las variables esta relacionada estrechamente con la variable *Response* ni con ninguna otra. Por tanto, mantenemos todas las variables que tenemos.

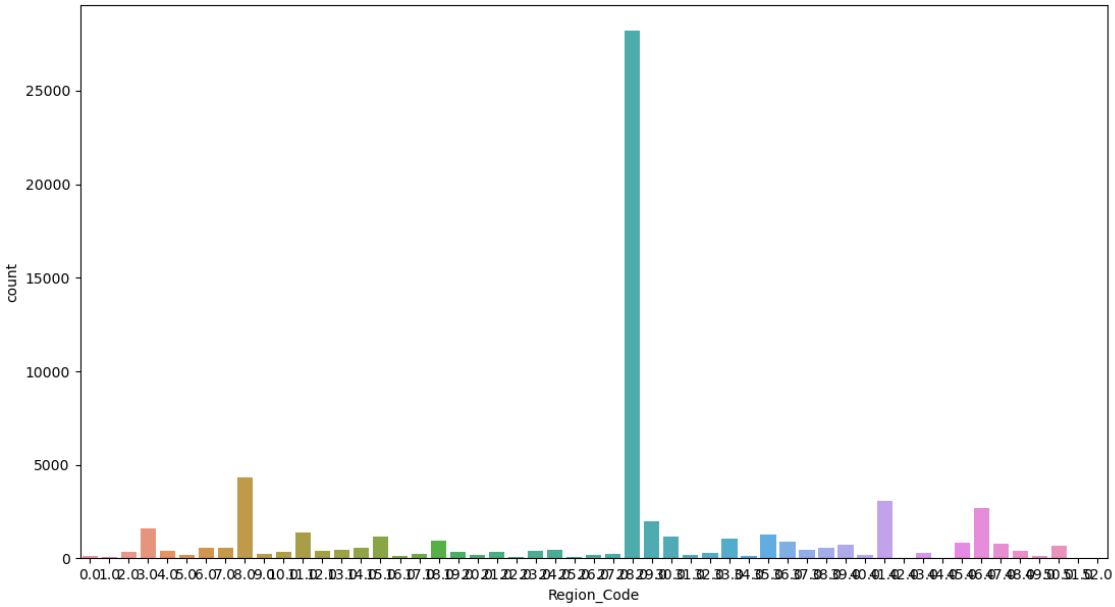


Figura 5.5: Región

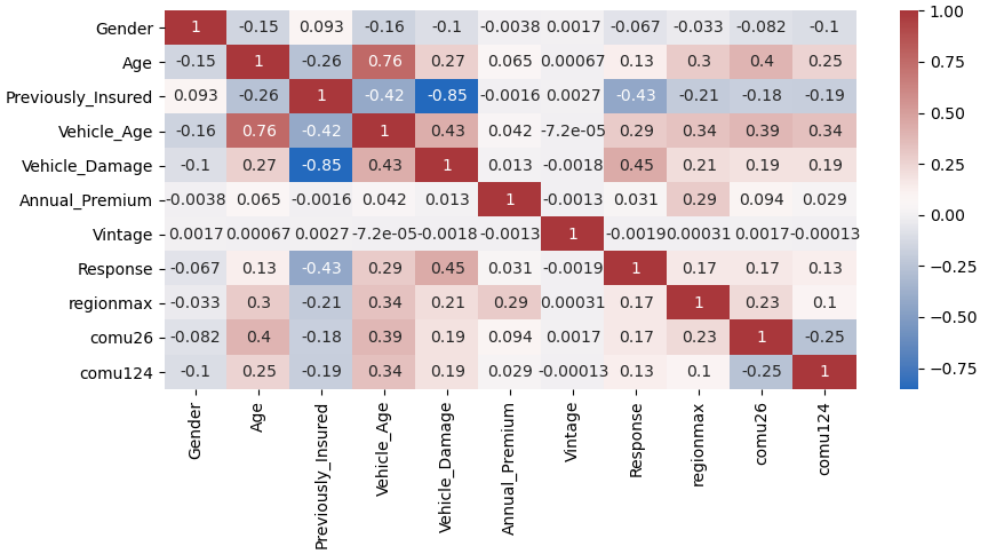


Figura 5.6: Matriz de correlación

Damos por finalizado el análisis de la base de datos. Realizamos una transformación de los datos por medio de la estandarización MinMax de manera que todos los datos nos quedan comprendidos entre los valores [0,1] y comenzamos con la clasificación.

5.3. Metodología general

Una vez hemos trabajado el conjunto de datos para comprender su forma, podemos comenzar a aplicar las técnicas para elaborar los distintos modelos. Trabajaremos con cuatro algoritmos de clasificación: árboles de decisión, regresión logística, *Random Forest* y *Adaboost*, y el procedimiento que seguiremos será el mismo para los cuatro. Los pasos que seguiremos serán:

- Establecer el clasificador que vamos a utilizar.
- Fijar las métricas que vamos a considerar para nuestro modelo. En nuestro caso veremos exactitud, preciisión, recall, F2-score y coeficiente de Matthews (MCC). Aunque el F1 score es una medida más estándar, hemos optado por F2-score por la importancia que tiene el recall sobre la precisión en este caso concreto.
- Aplicar una validación cruzada sobre el conjunto de datos para obtener las métricas que resultan del modelo en cada iteración y poder hacer luego un promedio que será nuestro resultado. Así conseguimos los valores para una clasificación de los datos desbalanceados.
- Ajustar el algoritmo en cuestión para que tenga en cuenta el peso de la clase minoritaria, aplicando una de las técnicas de refinamiento de algoritmos. Obtenemos los valores para una clasificación ajustando los algoritmos.
- Aplicar las técnicas de remuestreo: oversampling, undersampling y SMOTETomek. En este caso se aplicará la técnica sobre las particiones destinadas a entrenar el modelo, de manera que la partición de prueba mantendrá el desbalanceo original para dar una valoración más precisa de cómo actuará el modelo.

En los apartados siguientes analizaremos los resultados obtenidos para cada uno de lso algoritmos con las distintas técnicas.

5.4. Clasificación de datos limpios

En esta sección vamos a ver cómo se comportan distintos algoritmos al intentar clasificar los datos con el desbalanceo original y sin aplicar ninguna técnica. Nos servirá como guía para poder entender los resultados que obtendremos más adelante y para comprender el problema que se da en la clasificación de este tipo de conjuntos de datos.

En la tabla 5.1 vemos señalados en azul los mejores valores obtenidos para cada una de las métricas. Si nos fijamos en la exactitud puede dar la sensación de que todos los clasificadores hacen una clasificación similar y que además, esta es bastante buena ya que todos presentan en torno a un 84 % de acierto. Sin embargo, observamos que el de recall no supera ni un 30 %, por lo que la mayoría de la clase minoritaria ni siquiera es reconocida como tal. Este es el principal problema del desbalanceo

Algoritmo	exactitud	precision	recall	F2-score	MCC
Árbol dec.	0.8398±0,0009	0.5300±0,0108	0.2029±0,0420	0.2307±0,0423	0.2549±0,0236
Reg.Log	0.8357±0,0013	0.4952 ±0,0090	0.1425 ±0,0042	0.1662±0,0047	0.1992±0,0059
Random For.	0.8418±0,0008	0.5482 ±0,0068	0.1950 ±0,0090	0.2238±0,0093	0.3639±0,1114
AdaBoost	0.8414±0,0012	0.5352±0,0048	0.2402±0,0051	0.2700±0,0053	0.2827±0,0054

Tabla 5.1: Clasificación de datos limpios

de clases. Obtenemos una buena exactitud por la cantidad de instancias de clase mayoritaria que existen, pero realmente está actuando de manera pobre con respecto a la clase minoriaria. Ninguno de estos modelos podría darse como bueno para una clasificación.

5.5. Refinamiento de algoritmos

Hemos visto ya como clasifican distintos clasificadores nuestra muestra original. A continuación vamos a aplicar un ajuste en los parámetros de alguno de ellos y veremos cómo de una manera muy simple, obtenemos una mejora en los resultados de la clasificación. Este ajuste se realiza sin modificar la base de datos, lo que haremos es añadir el parámetro *classweight = balanced* al clasificador en el momento de entrenar los datos. De esta manera, el clasificador utiliza los valores de la clase para ajustar automáticamente los pesos de manera inversamente proporcional a la frecuencia de cada clase en la muestra [20]. Adaboost no dispone de este parámetro, lo vemos en el resto para comparar en igual de condiciones las mejoras.

Algoritmo	exactitud	precision	recall	F2-score	MCC
Árbol dec.	0.7743±0,0015	0.4148±0,0018	0.9198±0,0014	0.7397±0,0011	0.5122±0,0015
Reg. Log.	0.7069±0,0014	0.3562 ±0,0015	0.9773 ±0,0015	0.7246±0,0013	0.4697±0,0016
Random For.	0.7702±0,0025	0.4109±0,0027	0.9278±0,0018	0.7412±0,0017	0.4906±0,0221

Tabla 5.2: Clasificación de datos limpios aplicando refinamiento.

En la tabla 5.2 podemos observar una mejora general respecto a los datos obtenidos en la clasificación original de la muestra que vimos en la tabla 5.1. Observamos que, aunque la exactitud y la precisión se vean afectadas reduciéndose, el nivel de recall alcanzado es muy superior al 90 %. También las medidas de rendimiento general indican una mejora en la clasificación global. Resaltamos que, aunque el recall sea más alto en el caso de la regresión logística, en el caso de esta clasificación de clase minoritaria no es de vital importancia con lo que aceptaríamos como mejor modelo el árbol de decisión ya que presenta mejores valores en general, teniendo también un buen recall.

5.6. Remuestreo

En esta sección vamos a aplicar distintas técnicas de remuestreo al conjunto de entrenamiento de manera que las clases estén balanceadas. Empezaremos duplicando instancias de la clase minoritaria mediante random oversampling, seguiremos con random undersampling reduciendo el número de entradas de la clase mayoritaria y finalizaremos aplicando una combinación de las técnicas de SMOTE y Tomek links.

5.6.1. *Oversampling*

Aplicamos random oversampling sobre cada conjunto de entrenamiento en las iteraciones de la validación cruzada consiguiendo que ambas clases tengan la misma cantidad de instancias (en torno a 255600), a base de aumentar la clase minoritaria creando clones de datos ya existentes.

Algoritmo	exactitud	precision	recall	F2-score	MCC
Árbol dec	0.7720$\pm$0,0042	0.4124$\pm$0,0041	0.9221$\pm$0,0051	0.7393$\pm$0,0012	0.5106$\pm$0,0026
Reg.Log	0.7068$\pm$0,0013	0.3561 $\pm$0,0015	0.9773 $\pm$0,0015	0.7246$\pm$0,0013	0.4696$\pm$0,0016
Random For.	0.7687$\pm$0,0017	0.4093$\pm$0,0021	0.9293$\pm$0,0017	0.7410$\pm$0,0018	0.5104$\pm$0,0024
AdaBoost	0.7640$\pm$0,0034	0.4044$\pm$0,0034	0.9315$\pm$0,0022	0.7388$\pm$0,0019	0.5060$\pm$0,0032

Tabla 5.3: Clasificación de datos balanceados con random oversampling.

En la tabla 5.3 se muestran en azul los mejores valores para cada una de las métricas. Si comparamos estas con la tabla 5.1 vemos que se han conseguido mejorar el rendimiento de la clasificación en todos con todos los algoritmos. Apreciamos un valor de recall alto en todos los modelos donde destaca la regresión logística. Sin embargo, este modelo presenta el peor resultado a nivel de exactitud y precisión. Aunque algunas métricas se hayan visto empeoradas, ha sido en muy poca medida en comparación con la mejora que ha supuesto en la detección de la clase minoritaria que obtenemos con el recall. Además, el F2-score y el MCC que nos dan una información global del clasificador han mejorado notablemente.

5.6.2. *Undersampling*

Aplicando random undersampling sobre nuestra muestra, conseguimos que ambas clases tengan la misma cantidad de instancias (en torno a 50000), a base de disminuir la clase mayoritaria eliminando aleatoriamente instancias hasta equilibrar ambas clases.

Como venimos haciendo, la tabla 5.4 muestra en azul los mejores valores para cada una de las

Algoritmo	exactitud	precision	recall	F2-score	MCC
Árbol dec	0.7724±0,0050	0.4128±0,0061	0.9201±0,0061	0.7385±0,0014	0.5102±0,0037
Reg.Log	0.7067±0,0012	0.3561 ±0,0013	0.9777 ±0,0009	0.7247±0,0014	0.4697±0,0017
Random For.	0.7672±0,0030	0.4079±0,0030	0.9322±0,0024	0.7415±0,0018	0.5101±0,0028
AdaBoost	0.7633±0,0026	0.4037±0,0026	0.9321±0,0026	0.7388±0,0018	0.5055±0,0028

Tabla 5.4: Clasificación de datos balanceados con undersampling.

métricas. Observamos que con esta técnica obtenemos unos resultados muy similares a los que hemos visto para el random oversampling, presentando por tanto, una notable mejora con respecto a los valores de la clasificación inicial sin aplicar técnicas para mejorarla. Fijándonos en las métricas de F2-score y el coeficiente de Matthews, que nos dan una visión más global del clasificador podemos apreciar también esta mejora.

5.6.3. *Generación muestras sintéticas*

En esta sección vamos a ver los resultados obtenidos tras aplicar conjuntamente las técnicas de SMOTE y Tomek-Links a partir de la función de SMOTETomek de la librería de python *imbalanced-learn*. Este método combina la habilidad de generar muestras sintéticas de la clase minoritaria con SMOTE y la habilidad de Tomek Links de eliminar datos cercanos que puedan crear confusión. Obtenemos así una muestra de entrenamiento de alrededor de 68500 registros.

Algoritmo	exactitud	precision	recall	F2-score	MCC
Árbol dec	0.7761±0,0117	0.4161±0,0117	0.9035±0,0199	0.7317±0,0039	0.5063±0,0046
Reg.Log	0.7074±0,0015	0.3564±0,0016	0.9763 ±0,0019	0.7243±0,0014	0.4695±0,0016
Random For.	0.7720±0,0028	0.4125±0,0030	0.9224±0,0021	0.7395±0,0018	0.5108±0,0028
AdaBoost	0.7700±0,0050	0.4099±0,0052	0.9177±0,0071	0.7354±0,0021	0.5060±0,0033

Tabla 5.5: Clasificación de datos balanceados con SMOTETomek.

Volvemos a mostrar en azul los mejores resultados para cada métrica. Vemos que con esta técnica también obtenemos una mejora en la clasificación en la misma medida que hemos ido viendo con el resto de técnicas. Conseguimos un buen recall y un incremento del rendimiento global reflejado en las métricas de F2-score y coeficiente de Matthews.

5.7. Análisis de resultados

En las secciones anteriores hemos visto los resultados de utilizar las técnicas explicadas. Hemos podido observar una clara mejora en el nivel de predicción de la clase minoritaria tras aplicarlas. Sin embargo, resulta interesante ver que, en realidad, la curva precisión-recall apenas se ve alterada tras realizar estas mejoras. A continuación vamos a ver qué es lo que sucede en el nivel de clasificación para que los resultados se vean modificados sin influir en esta curva. Para ello, vamos a mostrar la curva precision-recall de cada algoritmo y el nivel de predicción en cada una.

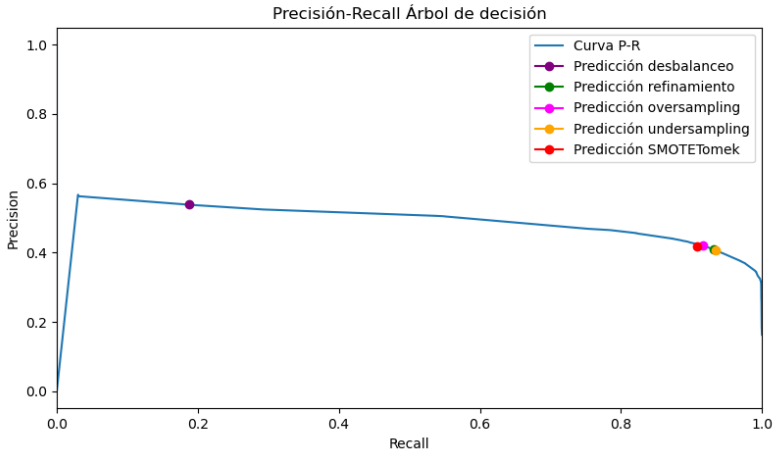


Figura 5.7: Predicciones árbol decisión

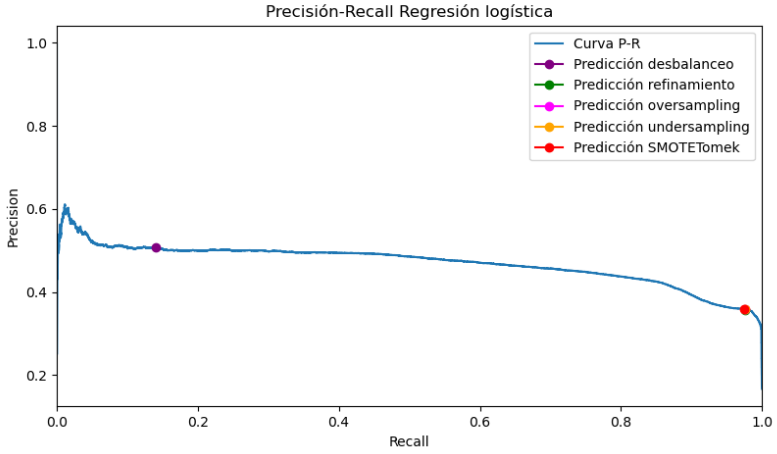


Figura 5.8: Predicciones regresión logística

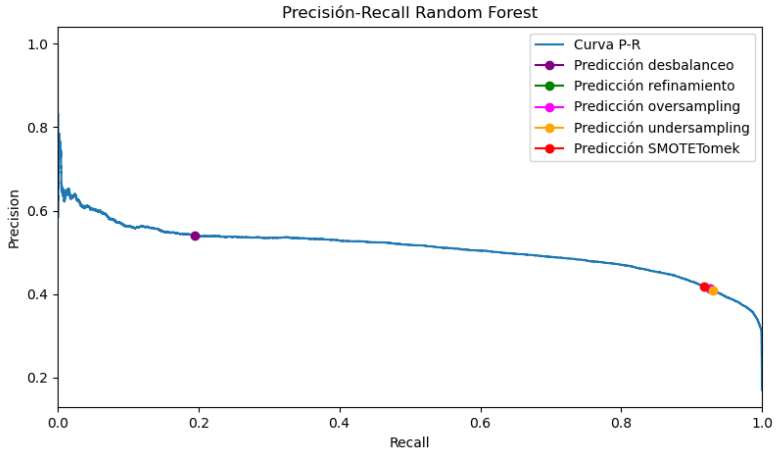


Figura 5.9: Predicciones random forest

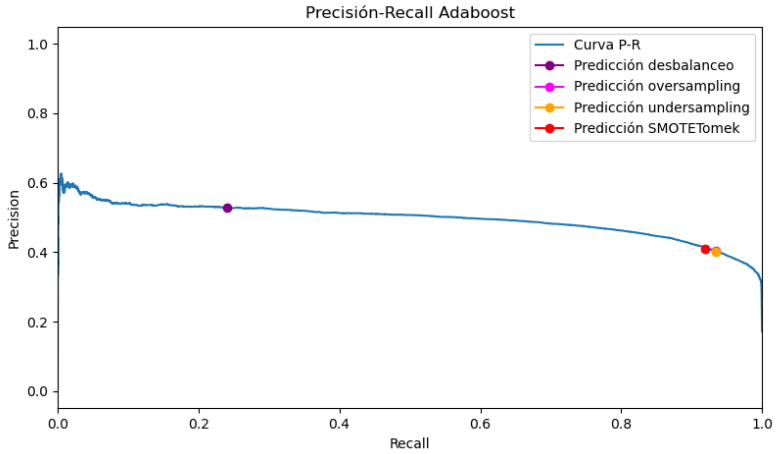


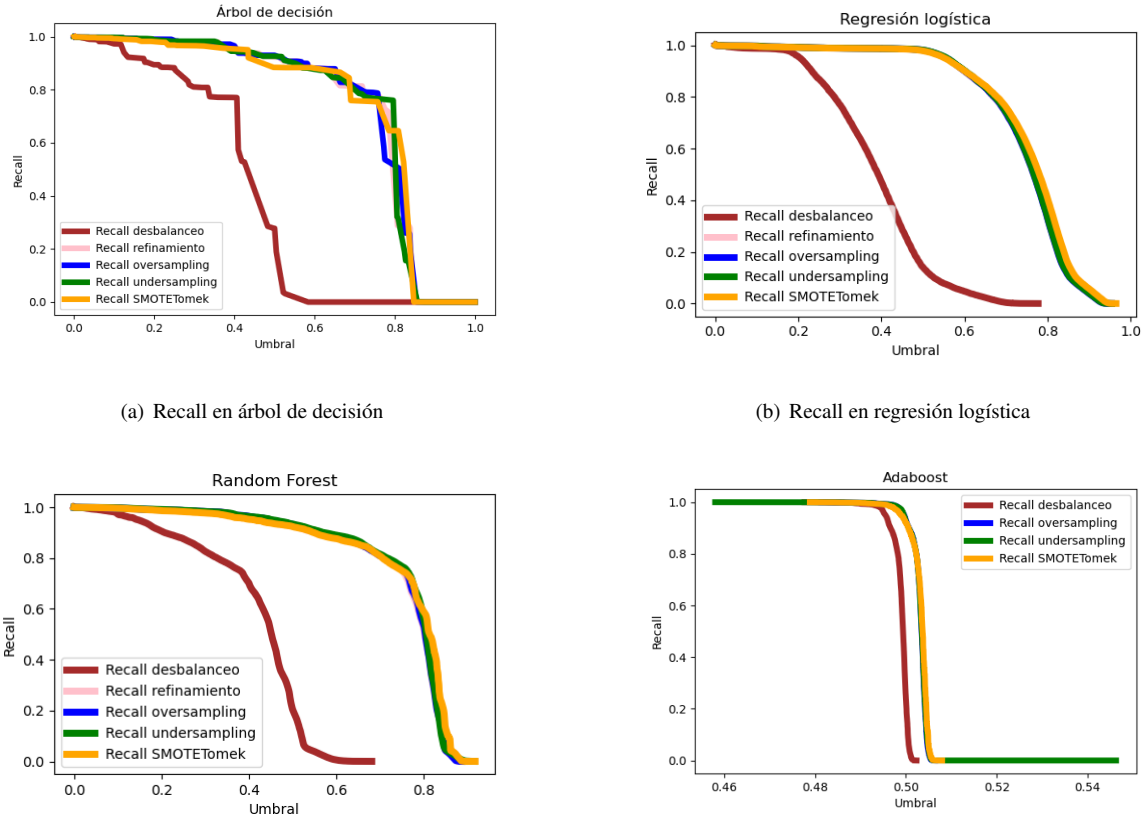
Figura 5.10: Predicciones adaboost

Para poder entender bien el significado de las gráficas obtenidas en las Figuras 5.7, 5.8, 5.9 y 5.10, primero debemos profundizar en el proceso de cálculo de la curva precisión-recall. La clave está en la probabilidad con la que se decide que un elemento pertenece a una clase. Realmente, durante el proceso de clasificación no se asigna una etiqueta directamente sino una probabilidad. En el caso de la regresión logística, por ejemplo, viene determinada directamente por la propia función. En otros casos como el *random forest*, se predice a partir del número de árboles que dicen cada clase. La curva se construye calculando los puntos (recall, precisión) para cada una de estas probabilidades, que se denominan umbrales. De esta forma, la unión de los puntos genera la curva precisión-recall. Naturalmente, cuanto mayor sea la muestra, más suave será la curva al tener cambios menos bruscos por estas las probabilidades más cerca unas de otras.

Una vez comprendido este mecanismo, procedemos a un análisis general de las gráficas para ilustrar en qué han influido las técnicas aplicadas. Para empezar destacamos que las predicciones

son prácticamente iguales en todas las técnicas. No debe sorprendernos después de haber visto los resultados obtenidos en las secciones anteriores donde obteníamos resultados similares para cada métrica. Mientras todas estas se agrupan al lado derecho de la gráfica, la predicción con los datos desbalanceados, se sitúa más a la izquierda. Esto se traduce en que hemos conseguido que, mediante el entrenamiento de los modelos con las técnicas empleadas, se haya encontrado el umbral que consigue maximizar la relación precision-recall.

Además, podemos observar una mejora en la calidad de la predicción a partir de las siguientes gráficas del recall.



(a) Recall en árbol de decisión

(b) Recall en regresión logística

(c) Recall en Random Forest

(d) Recall en Adaboost

Figura 5.11: Recall en distintos algoritmos

En la Figura 5.11 podemos ver como va aumentando el nivel de recall a medida que el umbral de clasificación va disminuyendo. Es decir, cuanto menos 'exigentes' somos para aceptar un dato en la clase minoritaria, obtenemos mejor recall. Observamos que salvo para el caso de Adaboost, obtenemos una gran mejora en el recall en el sentido de que conseguimos un recall alto en un umbral más alto y por tanto mejor detección de la clase minoritaria ya que las estará acertando con una mayor probabilidad.

6

Conclusiones y limitaciones

6.1. Conclusiones

Tras haber realizado este trabajo conseguimos un conocimiento sobre el área de la clasificación de bases de datos desbalanceados, sus problemas y la importancia de encontrar técnicas que mejoren sus resultados. Además hemos aprendido por qué un clasificador no debe ser medido por su nivel de exactitud y por qué este puede ser engañoso.

Después de realizar los experimentos para clasificar la muestra y aplicar los distintos métodos explicados podemos observar que obtenemos mejores resultados en la mayoría de las métricas de nuestro interés y una mejora global en el rendimiento de cada clasificador. Si bien apreciamos una mejora del mismo calibre con todas las técnicas, no podemos extrapolar este resultado a todos los casos de clasificación ya que cada muestra es una situación única.

El entrenamiento con datos balanceados permite que los modelos aprendan más fielmente las funciones de densidad de probabilidad de las clases y, en consecuencia, van a definir mejor cuáles son los umbrales que maximizan la relación precisión-recall.

6.2. Limitaciones y trabajo futuro

En este trabajo hemos visto una pequeña introducción al área de la clasificación de bases de datos desbalanceadas. Si bien es cierto que hemos visto que para este tipo de bases de datos es muy difícil dar una clasificación realmente buena, podemos aplicar ciertas técnicas que mejoran la calidad del modelo.

Aún quedarían muchos detalles que investigar, por ejemplo, hacer una selección de las variables más importantes para quitar posible ruido o hacer que variables más importantes no pasen desapercibidas entre la multitud. Además, estudiar algoritmos más eficientes para cada caso en concreto y técnicas de remuestreo más complejas y aplicarlo en bases de datos que presenten más desequilibrio entre las clases ya que en esta teníamos un desbalanceo de 1:5 aprox. Sería interesante también ver su comportamiento a nivel clasificación multiclase.

Bibliografía

[1] "Clasificación desbalanceada." https://ichi.pro/es/umbral-optimo-para-clasificacion-desequilibrada-973 Accedido 03-2021.

[2] "Base de datos." https://www.kaggle.com/arashnic/imbalanced-data-practice. Accedido 02-2021.

[3] "Machine-learning." https://www.iberdrola.com/innovacion/machine-learning-aprendizaje-automatico. Accedido 03-2021.

[4] "Machine-learning." https://empresas.blogthinkbig.com/que-algoritmo-elegir-en-ml-aprendizaje/. Accedido 03-2021.

[5] "Clasificación." https://docs.microsoft.com/es-es/dotnet/machine-learning/resources/tasks. Accedido 03-2021.

[6] "Validación cruzada." https://www.cienciadedatos.net/documentos/30_cross-validation_oneleaveout_bootstrap. Accedido 03-2021.

[7] "Árboles de decisión." https://www.maximaformacion.es/blog-dat/que-son-los-arboles-de-decision-y-para-que-sirven/. Accedido 03-2021.

[8] C. M. Salcedo Poma, "Estimación de la ocurrencia de incidencias en declaraciones de pólizas de importación.," Surprise library: http://surpriselib.com/.

[9] "Multiclasificadores." https://machinelearningparatodos.com/cual-es-la-diferencia-entre-los-metodos-d Accedido 03-2021.

[10] "Adaboost." https://algotech.netlify.app/blog/xgboost/. Accedido 03-2021.

[11] "Métricas." https://empresas.blogthinkbig.com/ml-a-tu-alcance-matriz-confusion/. Accedido 03-2021.

[12] "Coeficiente de matthews." https://ichi.pro/es/coeficiente-de-correlacion-de-matthews-cuando-usarlo-y Accedido 04-2021.

[13] "Problema del desbalanceo." https://machinelearningmastery.com/imbalanced-classification-is-hard/. Accedido 03-2021.

[14] "Técnicas de remuestreo." https://machinelearningmastery.com/random-oversampling-and-undersampling-fo Accedido 03-2021.

[15] "Smote." https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/. Accedido 03-2021.

[16] "Tomek-links." https://www.kdnuggets.com/2016/08/learning-from-imbalanced-classes.html/2. Accedido 04-2021.

[17] L. Peng, H. Zhang, B. Yang, and Y. Chen, "A new approach for imbalanced data classification based on data gravitation," *Information Sciences*, vol. 288, p. 347–373, 12 2014.

[18] "Python." https://www.cienciadedatos.net/documentos/py06_machine_learning_python_scikitlearn.html. Accedido 06-2021.

[19] "Imblearn." https://pypi.org/project/imbalanced-learn/. Accedido 06-2021.

[20] "Peso balanceado." https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. Accedido 03-2021.